开篇词 | 想吃透Go并发编程，你得这样学！

2020-10-12 晁岳攀

Go 并发编程实战课 进入课程 >

讲述：安晓辉
时长 12:22 大小 11.33M

你好，我是晁岳攀，网名鸟窝。之前我在微博研发平台架构中心担任资深架构师，同时也是微服务框架 rpcx 的作者，欢迎来到“Go 并发编程实战课”。

并发编程，为什么选 Go？

为什么要学 Go 并发呢？我想先和你聊聊我和 Go 结缘的经历。

作为一位老程序员，我在清华同方、摩托罗拉、Comcast 等公司，一直使用 Java 做项目开发。但是后来，我毅然抛弃了十几年的 Java 编程经验，投入到了 Go 语言的怀抱，为什么呢？

一句话，我被 Go 的简单高效所打动。它不仅部署方便，自带完善的工具链，特别是 Go 在处理并发场景上表现出的独特性能，更是让我着迷。

我们知道，Java 语言的编码非常繁琐，为了应用设计模式而做了大量的冗长设计，而 Go 就不一样了。它提供了便利的并发编程方式，简简单单的 Go 语句，就可以创建多个 goroutine 执行并发任务。而且，Go 还提供了独特的 Channel 类型，很容易实现 goroutine 之间的数据交流。所以，Go 并发编程入门很容易，即使是初学者，要写一个使用 goroutine 异步输出“Hello World”的例子，也可以不费吹灰之力。

不过，和其他语言相比，Go 微服务治理框架的发展还是比较晚的。当阿里出品的 Java 微服务框架 Dubbo 被广泛应用时，Go 生态圈还没有微服务框架。

于是，四五年前，为了填补 Go 生态圈微服务化的缺失，我就用 Go 开发了一个微服务的框架 rpcx。它既有类似标准 rpc 库的易用特点，又包含了非常丰富的服务治理的功能。而且，根据benchmark 测试，rpcx 有着数一数二的性能，很多互联网企业（比如马蜂窝、百度等）都在使用。

在微博的四年时间里，我使用 Go 参与开发多个基础架构系统，并负责中国版权链、微博下一代的 Redis 集群系统、数据库资源云等系统的设计和开发工作。在多年的实战中，我遇见过各种各样的并发难题，积累了大量的高并发高吞吐的服务器开发经验，也梳理了一整套并发编程的知识体系。

2019 年，astaxie（谢孟军）邀请我在 Gopher China 大会上做一个关于 Go 并发编程的分享。我准备了一份 120 页的PPT，全面地介绍了 Go 并发编程的基础内容，包括基本并发原语、扩展并发原语和 Channel 等。会后，现场的观众都说干货满满，希望我能提供无删改版的 PPT。

后来，在 Go 爱好者的强烈要求下，我又在滴滴举办了一场 Go 并发编程的培训，详细地分享了我的并发编程心得和经验，包括各种并发原语的基本用法和实现机制。

结合我自己的开发经验，以及这些年的技术分享经历，我真切地感受到了这一点：**Go 并发编程的重要性不容置疑。只要是使用 Go 开发的大型应用程序，并发是必然要采用的技术。**

但同时，我也了解到，很多人想要学习 Go 并发编程，却不知道该从何学起，也不知该如何精进。

学习 Go 并发编程，有哪些困难？

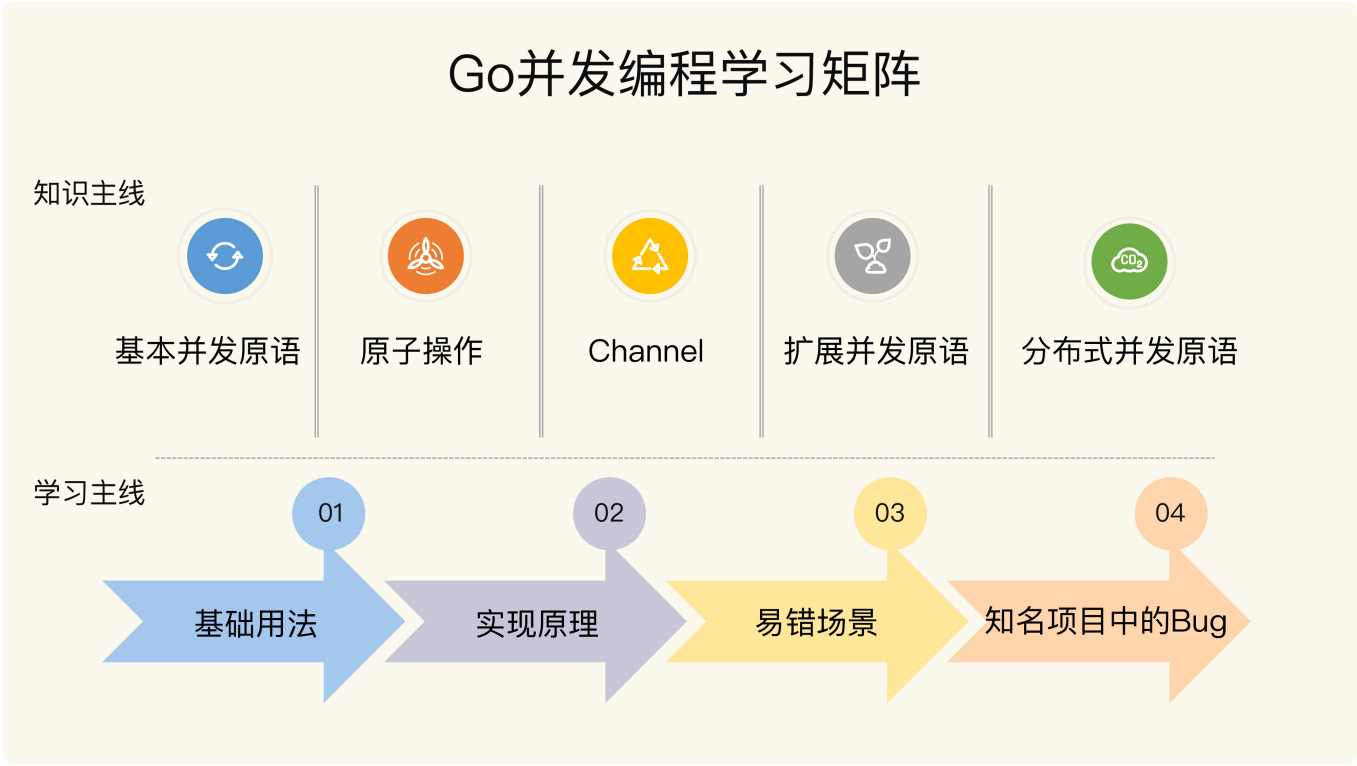
那学习 Go 并发会有哪些困难呢？我总结了一下，主要是有 5 大问题。

1. 在面对并发难题时，感觉无从下手，不知道**该用什么并发原语来解决问题**。
2. 如果多个并发原语都可以解决问题，那么，**究竟哪个是最优解呢**？比如说是用互斥锁，还是用 Channel。
3. **不知道如何编排并发任务**。并发编程不像是传统的串行编程，程序的运行存在着很大的不确定性。这个时候，就会面临一个问题，**怎么才能让相应的任务按照你设想的流程运行呢**？
4. 有时候，按照正常理解的并发方式去实现的程序，结果莫名其妙就 panic 或者死锁了，**排查起来非常困难**。
5. **已知的并发原语都不能解决并发问题**，程序写起来异常复杂，而且代码混乱，容易出错。

每一位刚入门 Go 的程序员，在深入学习 Go 语言的时候，尤其是面对 Go 并发编程的时候，都会遇到这些问题。那么，具体该怎么学呢？

怎么提升 Go 并发编程能力？

学习这件事儿，最怕的就是不成体系，即使知识点之间是彼此独立的，也必定存在着联系。我们要做的，就是找出逻辑关系，拎出知识线。我认为，**关于 Go 并发编程，有两条主线，分别是知识主线和学习主线**。具体是啥意思呢？可以看下面的这张知识地图。



从图中可以看到，在知识主线层面，这门课程的核心内容设计了 5 个模块：

基本并发原语：在这部分，我会介绍 Mutex、RWMutex、Waitgroup、Cond、Pool、Context 等标准库中的并发原语，这些都是传统的并发原语，在其它语言中也很常见，是我们在并发编程中常用的类型。

原子操作：在这部分，我会介绍 Go 标准库中提供的原子操作。原子操作是其它并发原语的基础，学会了你就可以自己创造新的并发原语。

Channel：Channel 类型是 Go 语言独特的类型，因为比较新，所以难以掌握。但是别怕，我会带你全方位地学习 Channel 类型，你不仅能掌握它的基本用法，而且还能掌握它的处理场景和应用模式，避免踩坑。

扩展并发原语：目前来看，Go 开发组不准备在标准库中扩充并发原语了，但是还有一些并发原语应用广泛，比如信号量、SingleFlight、循环栅栏、ErrGroup 等。掌握了它们，就可以在处理一些并发问题时，取得事半功倍的效果。

分布式并发原语：分布式并发原语是应对大规模的应用程序中并发问题的并发类型。我主要会介绍使用 etcd 实现的一些分布式并发原语，比如 Leader 选举、分布式互斥锁、分布式读写锁、分布式队列等，在处理分布式场景的并发问题时，特别有用。

想吃透 Go 并发编程，你得这样学！

基本并发原语

01 Mutex：如何解决资源并发访问问题?

02 Mutex：庖丁解牛看实现

03 Mutex：4 种易错场景大盘点

04 Mutex：骇客编程，如何扩展额外功能?

05 RWMutex：读写锁的实现原理及避坑指南

06 WaitGroup：协同等待，任务编排利器

07 Cond：条件变量的实现机制及避坑指南

08 Once：一个简约而不简单的同步原语

09 Map：如何实现线程安全的 Map 类型?

10 Pool：性能提升大杀器

11 Context：信息穿透上下文

原子操作

12 atomic：要保证原子操作，一定要使用这几种方法

Channel

沿着这条知识主线，我会带你建立起一个丰富的并发原语库。你可以把并发问题当成一个强大的敌人，而这些并发原语，就是我们的武器。每一种并发原语都有它的用处，你只有知道足够多的并发原语，才能灵活地应对各种场景。

那具体怎么掌握这些武器呢？课程的每一个模块都是独立的，它们之间没有任何依赖问题，你可以结合自己的实际情况，有重点地进行学习。如果你对 Channel 类型不是太熟悉，就可以先看 Channel 这个模块的内容；如果你已经非常熟悉标准库的并发原语了，就可以看看扩展并发原语和分布式并发原语的内容。

同时，在学习主线层面，主要是**四大步骤，包括基础用法、实现原理、易错场景、知名项目中的 Bug**。每一个模块，我都会带着你按照这四个步骤来学习，目的就是带你熟知每一种并发原语的实现机制和适用场景。

Go 中有一个大的方向，就是任务编排用 Channel，共享资源保护用传统并发原语。在刚开始学习时，你可以基于这个原则去选择相应的并发原语，这是没错的。但是，如果你想要在 Go 并发编程的道路上向前走，就不能局限于这个原则。

实际上，针对同一种场景，也许存在很多并发原语都适用的情况，但是一定是有最合适的那一个。所以，你必须非常清楚每种并发原语的实现机制和适用场景，千万不要被网上的一些文章误导，万事皆用 Channel。

而且，你还可以深入学习下 Go 并发原语的源代码。你会发现很多独到的设计，比如 Mutex 为了公平性考量的设计、sync.Map 为提升性能做的设计，以及很多并发原语的异常状况的处理方式。尤其是这些异常状况，常常是并发编程中程序 panic 的原因。

所以，如果你能深入了解这些并发原语的实现，不但会提高你的编程能力，还能让你避免在开发中踩并发问题的坑。这个时候，你就达到精通的程度了。

如果没有做过大型并发项目，你可能还不太清楚并发原语的重要性。那么，我建议你先阅读一下课程中介绍的知名项目中犯的错，这也是这门课里我特别设计的一部分内容。通过理解这些 Go 大牛们犯的错误以及解决方案，你就可以积累一套避坑指南和应对之道。

有了这两条线的学习，我们就从广度和深度上掌握了 Go 并发编程的知识点。这些是不是就足够了呢？我们还可以更进一步，你要有野心能够创造出自己需要的并发原语。

这里的创造有两层含义。第一层是对既有的并发原语进行组合，使用两个、三个或者更多的并发原语去解决问题。比如说，我们可以通过信号量和 WaitGroup 组合成一个新的并发原语，这个并发原语可以使用有限个 goroutine 并发处理子任务。第二层含义是“无中生有”，根据已经掌握的并发原语的设计经验，创造出合适的新的并发原语，以应对一些特殊的并发问题。比如说，标准库中并没有信号量，你可以自己创造出这个类型。

达到了这一层，那就不得了了，可以说你对 Go 并发原语的掌握已经出神入化了。那想要达到这个程度是不是很难呢？确实不容易，不过我相信，如果你仔细学习了我们课程里的

每一节课，心里牢牢地锚定 3 个目标：建立起一个丰富的并发原语库；熟知每一种并发原语的实现机制和适用场景；能够创造出自己需要的并发原语。达到了这 3 个目标，你就可以轻松地应对各种并发问题了。甚至可以说，你几乎能站在 Go 并发编程的顶端，成为大牛中的一员。

最后，我想说的是，Go 并发编程的世界确实纷繁复杂，涉及到的内容非常多。你可以把它看作是一个江湖，如果你想拥有极强的作战力，就要拥有足够多的武器，并且修炼内功。这门课，就是你的修炼山洞，我准备了应有尽有的宝藏，等待着你来挖掘。

修炼的过程中，最好有人和你并肩而行，共同成长。欢迎你把这门课分享给你的朋友或同事，和他 / 她一起提升并发编程的功力。

55 人觉得很赞 | 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

下一篇

精选留言 (28)

写留言

五河士稻
2020-10-13

没啥特别要说的，就看这发型买了。

展开

4 63

Alexdown
2020-10-12

请问这个专栏内容的广度和深度上与您在gopherchina分享的内容和滴滴培训所授内容有什么区别？是其子集还是并集？谢谢！

展开

作者回复: 这个相当于重新整理和扩充，补充一些内容，细化了知识点，补充了例子和错误案例

11

端贺
2020-10-12

同老师一样，目前从事Java开发，正在转Go。在没有接触Go的时候，认为自己会用Java一辈子，接触了Go语言，感觉相见恨晚。
希望在老师的指导下，可以更深入了解Go语言，写出更优秀的代码。

展开

作者回复: 加油，期待你学完之后成为go并发专家

5

南有乔木
2020-10-13

老师您好，第一时间买了您的课程，请问团队里很多java的开发试图拉我回到java阵营，列举了很多理由，比如没有go没有spring这种依赖注入的框架等，想问一下精通go却不懂java是否会影响以后的发展？

作者回复: 很多公司的团队已经使用go做主要语言开发了，所以有时间多了解一些go的知识没有坏处。学语言不是非黑即白的问题。以目前的工作语言做主力，再兼顾一两门其它语言，均衡发

展

4

光明
2020-10-15

来子一个GO 小白的打卡。 虽然啥都不知道，先打卡 。

作者回复: 学完就成专家了

2

朱亚光
2020-10-13

只会用waitgroup 的我， 期待搞定go并发编程

展开

作者回复: Mutex,channel必会，期待一个月后的你并发源于全掌握

1

Gopher
2020-10-13

看大佬博客“长大”的，前来支持一波

展开

1

chapin
2020-10-12

头排，支持鸟窝大佬。

展开

2 1

all
2020-10-26

老师的发型和网名一样有型！

展开

the7th
2020-10-23

我就很奇怪，为什么还是java，C++

展开

阿豪
2020-10-22

打个卡，立个flag

展开

houseme
2020-10-20

打卡

展开

Jasper
2020-10-20

入门过基础，也写了简单的项目，始终没有什么体会，这门课来得刚刚好哈哈哈

小炭
2020-10-19

Go 的并发性能是不是就一定Java好，他们之间最大的区别是什么

oCupJS
2020-10-18

打卡

展开

kylexy_0817
2020-10-16

老师好。Go语言之于Java，唯一的缺点就是在编写复杂业务代码时层次结构够面向对象吧？

作者回复: 确定这是缺点？ ������可以看看Rob Pike关于oop和设计模式的思考

欧阳安
2020-10-15

233，我正在看rpcx的源码，边看边理解边动手敲，太多底层的细节优化了..

作者回复: 加油

大力水手
2020-10-15

打卡1 经常看你的博客、所以看到就直接订阅了、博客用的哪个框架？感觉代码段颜色很正。

桑成龙
2020-10-14

打卡

展开

Zoctopus
2020-10-14

学习go的时候经常会搜到您的博客^_^

展开